Software Engineering Project

The intent is for you to use an existing software development process to build a sample software system. You need to use one of the agile development methodologies for this project, in specific, Scrum. You will start with a small piece of software, and it will continue to grow over time using a series of sprints. Each sprint is composed of a plan, build, test, and review cycle. In your team, you are going to assume different roles which you may subdivide amongst you for role playing. For this methodology, you may need to maintain a Kanban board to keep track of the story you are working on, such as the todo, in progress, and done.

Initial Milestone (2 weeks):

A feasibility study and plan of your project which includes:

1. Your team names and emails
2. A two page description that shows you understand the Scrum methodology, and how it is conducted.
3. The possible client (could be hypothetical), that you are building this project for.
4. Overview of your system.
5. The product backlog which will include all the requirements you can think of from start to end (including functional and non-functional).
6. A high level architecture of the system based on your product backlog, which must be done using UML component diagrams.
7. A set of use cases using use case diagrams.
8. Any identified risks.
9. Conclusion.

First Sprint (about 2 weeks):

This milestone will include the following:

1. Planning of your first sprint.
2. The sprint backlog (the prioritized set of features you will work on during this sprint).
3. A development of this sprint.
4. Testing of the sprint.
5. A review of the status of the sprint.
6. A release of the increment with a demo and source code.
7. A sprint retrospective (what went well, what could be improved, what you will be doing for the next sprint).
8. A refined object oriented design for the features you developed (including any class diagram, sequence diagram, etc).

Deliverable: recorded demo of the software, source code, and report including your retrospective and refined design.

Second Sprint (about 2 weeks each):

You are going to repeat steps 1-8, and submit the same deliverables.

Final Milestone (2 weeks):

You will demonstrate your complete system in operation, and a handover of the completed system, along with all the artefacts:

- The detailed architecture.
- The detailed design.
- The refactored source code.
- Plan for future expansion.

All teams must use Github, and use UML.

SAMPLE PROJECT

Problem Description:

What shows below is just a sample of a project that you could be working on. You are encouraged to pick your own project, and not necessarily work on this specific project.

You are expected to engineer a software capable of controlling vehicles over a wireless network. Vehicles will be equipped with embedded software that can interface with a set of roadside sensors. Sensors are limited to: stop signs, pedestrian crossings, mobile warning devices (such as the ones used by construction workers), and RADAR devices alongside roads. The embedded software within vehicles also interfaces with a set of actuators within the car itself. Actuators are devices that convert various signals into mechanical actions that can slow down a car for example.

The embedded software should also interface with a traffic and weather database. The traffic and weather databases are continuously updated with information related to traffic and weather conditions at specific locations identified by a GPS system. Each vehicle is also equipped with a GPS system that can identify its exact location.

Very High Level Requirements:

1. Once a driver enters a vehicle, the vehicle engine cannot start until a valid driver identification is input into the system.
2. Through GPS tracking the vehicle's exactly location is reported to a database, along with its current speed.
3. A vehicle may undergo mandatory speed reduction to some mandated speed limit by the system. In all circumstances, a vehicle should be slowed down gracefully,

and under no circumstance, a vehicle shall not be slowed down to the mandated speed limit in less than five seconds.

4. The vehicle software shall access the traffic and weather database in order to receive alerts about heavy traffic and bad weather conditions. The vehicle should receive an alert for bad weather conditions once the vehicle is within ten kilometers of the weather condition itself. It should also receive alerts for heavy traffic conditions within a distance customizable by the vehicle itself.
5. The database classifies bad weather conditions according to three categories of severity. Each category of severity has an associated maximum speed limit that is customizable at the database itself.
6. An alert for bad weather should be signified at the vehicle in the form of an audible alarm. The alarm should indicate that the driver should slow down the vehicle to the mandated speed limit set at the database for this kind of weather condition. If the vehicle approaches two kilometers of the bad weather condition, and the car is still traveling at a higher speed limit, the audible alarm should indicate that the vehicle will be forcefully slowed down to the mandated speed limit within one kilometer distance from the end of the alarm message. The vehicle should abide by the mandatory speed reduction rules mentioned in (3). Such mandatory speed reduction is valid within the severe weather area, after which, the driver has full vehicle control again.
7. In the case of heavy traffic, an audible alert should be given to the driver, allowing the driver to manually take alternative routes.
8. Stop signs and pedestrian crossings emit signals signifying their presence. Once a stop sign is sensed, an audible alarm should be sounded in the vehicle.
9. Mobile warning devices also emit signals signifying their presence, but also emit a mandatory speed limit. Once a signal as such is sensed, the vehicle should undergo mandatory speed reduction, also abiding by the mandatory speed reduction rules mentioned in (3). Once the mobile warning device signal is not received any more, the driver has full control of the vehicle again.
10. A RADAR device capturing a vehicle violation immediately logs the violation in the form of a vehicle ID, the current speed, the required speed, and the date and time of violation to a database. An audible alarm should be given in the vehicle, and mandatory speed reduction initiated according to the rules mentioned in (3). Once the speed has been reduced to the desired limit, the driver has full control of the vehicle again.
11. A driver may deactivate automated control over the vehicle speed in case of emergency situations. However, such deactivation is immediately logged in a database.
12. The driver may request emergency assistance. A driver will then be sent an estimated time of arrival of the emergency crew within a maximum of ten minutes from the request.
13. An appropriate graphical user interface should be utilized.

What shows above is only a sample to show you the approximate high level scope of what you should be building.